IITP DABT API

서버 배포 설치 가이드

문서 버전: 1.0.0
작성일: 2025-10-28

(주)스위트케이

문서 History

버전	날짜	변경 내용	작성자
1.0	2025-10-24	최초 작성	(주)스위트케이

목차

1. 개요

IITP API 서버는 장애인 관련 데이터를 제공하는 RESTful API 서버입니다.

- 장애인 관련 통계 및 POI(Point of Interest), 편의시설 정보 등

본 문서는 빌드, 서버 배포, 운영에 대한 절차 및 정보를 제공합니다.

1.1 기술 스택

기술	버전	설명
Java	21	Spring Boot 3.x 요구사항
Spring Boot	3.2.5	웹 애플리케이션 프레임워크
PostgreSQL	16+	PostGIS 확장 지원
Gradle	8.13	빌드 도구
Jasypt	-	민감정보 암호화
Python	3.7+	OpenAPI 문서 그룹핑

2. 시스템 요구사항

2.1 빌드 환경 요구사항

빌드 및 개발을 위한 최소 요구사항입니다.

항목	요구사항	비고
OS	Linux (Ubuntu 22.04+), Windows 10+	빌드 및 배포
Java	JDK 21+	Spring Boot 3.x 요구사항
Gradle	8.13+	Gradle Wrapper 포함
Python	3.7+	OpenAPI 문서 그룹핑
메모리	최소 2GB RAM	빌드 시 필요
디스크	최소 10GB 여유공간	빌드 결과물 저장

2.2 운영 환경 요구사항

서버 운영을 위한 최소 요구사항입니다.

항목	요구사항	비고
OS	Linux (Ubuntu 22.04+), CentOS 8+	서버 운영
Java	JDK 21+	JRE 21+
메모리	최소 2GB RAM	운영 시 필요
디스크	최소 5GB 여유공간	로그 및 데이터
네트워크	28080 포트 오픈	API 서버 포트
PostgreSQL	16+	PostGIS 확장 지원

3. 빌드 환경 준비

빌드에 필요한 환경을 설정합니다.

3.1 Java 환경 설정

Java 21 이상이 설치되어 있는지 확인하고 환경변수를 설정합니다.

```
# Java 버전 확인
java -version

# JAVA_HOME 환경변수 설정 (필요시)
export JAVA_HOME=/usr/lib/jvm/java-21-openjdk
export PATH=$JAVA_HOME/bin:$PATH
```

3.2 Gradle 환경 설정

Gradle 래퍼를 사용하여 빌드 환경을 준비합니다.

```
# Gradle 래퍼 권한 설정
chmod +x gradlew

# Gradle 버전 확인
./gradlew --version

# 프로젝트 구조 확인
tree -L 2  # 또는 ls -la
```

3.3 Python 환경 설정

OpenAPI 문서 그룹핑을 위해 Python 환경을 설정합니다.

```
# Python 버전 확인
python3 --version

# 필요한 패키지 설치 (OpenAPI 문서 그룹핑용)
pip3 install requests
```

4. 데이터베이스 설정

PostgreSQL 데이터베이스 연결 및 PostGIS 확장 설정을 진행합니다.
PostgreSQL 버전은 16+ 이상을 권장합니다.

4.1 데이터베이스 연결 정보 설정

설치된 PostgreSQL 서버에 대한 연결 정보를 확인하고 설정합니다.

```
# 데이터베이스 연결 테스트
psql -h [호스트주소] -U [사용자명] -d [데이터베이스명]

# 예시: 개발 환경 연결 테스트
psql -h 192.168.60.146 -U iitp_dev -d iitp_dev
```

4.2 PostGIS 확장 확인 및 설치

공간 데이터 처리를 위해 PostGIS 확장을 설치합니다.

4.2.1 PostGIS 설치 여부 확인

```
# PostgreSQL에 접속하여 PostGIS 확장 확인
psql -h [호스트주소] -U [사용자명] -d [데이터베이스명] -c "SELECT PostGIS_version();"
```

확인 결과 예시:

```
postgis_version
----------------
 3.3 USE_GEOS=1 USE_PROJ=1 USE_STATS=1
```

PostGIS가 설치되지 않은 경우: 오류가 발생합니다.

4.2.2 PostGIS 설치 (Ubuntu/Debian)

PostGIS가 설치되지 않은 경우 다음 명령어로 설치합니다.

```
# 패키지 목록 업데이트
sudo apt update

# PostGIS 설치 (PostgreSQL 16 기준)
sudo apt install postgresql-16-postgis-3 postgresql-16-postgis-3-scripts

# 다른 버전의 경우:
# sudo apt install postgresql-14-postgis-3 postgresql-14-postgis-3-scripts
# sudo apt install postgresql-15-postgis-3 postgresql-15-postgis-3-scripts
```

4.2.3 데이터베이스에 PostGIS 확장 추가

PostGIS 패키지 설치 후, 각 데이터베이스에 PostGIS 확장을 추가해야 합니다.

```
# PostGIS 확장 생성
psql -h [호스트주소] -U [사용자명] -d [데이터베이스명] -c "CREATE EXTENSION postgis;"

# PostGIS 확장 확인
psql -h [호스트주소] -U [사용자명] -d [데이터베이스명] -c "SELECT PostGIS_version();"

# 추가 확장 (선택사항)
psql -h [호스트주소] -U [사용자명] -d [데이터베이스명] -c "CREATE EXTENSION postgis_topology;"
psql -h [호스트주소] -U [사용자명] -d [데이터베이스명] -c "CREATE EXTENSION postgis_raster;"
```

4.2.4 PostGIS 설치 확인

PostGIS 설치가 완료되면 다음 명령어로 확인합니다.

```
# PostGIS 버전 확인
psql -h [호스트주소] -U [사용자명] -d [데이터베이스명] -c "SELECT PostGIS_version();"

# 공간 참조 시스템 확인
psql -h [호스트주소] -U [사용자명] -d [데이터베이스명] -c "SELECT * FROM spatial_ref_sys LIMIT 5;'
```

4.3 거리 계산 방식 설정

거리 계산 방식을 설정합니다. application.yml에서 설정할 수 있습니다.

```yaml
app:
  distance-calculation:
    method: POSTGIS_SPHERE  # POSTGIS_SPHERE, POSTGIS_PLANAR, POSTGRESQL_EARTH
```

거리 계산 방식 비교:

방식	함수	정확도	성능	요구사항
POSTGIS_SPHERE	`ST_Distance_Sphere`	매우 좋음	좋음	PostGIS 확장
POSTGIS_PLANAR	`ST_Distance`	양호	매우 좋음	PostGIS 확장
POSTGRESQL_EARTH	`earth_distance`	좋음	좋음	PostgreSQL 기본

5. 환경별 설정 및 암호화

환경별 설정 파일 구조와 민감정보 암호화 설정을 진행합니다.

5.1 환경별 설정 파일 구조

환경별로 분리된 설정 파일들의 구조입니다.

```
src/main/resources/
├── application.yml              # 공통 설정
├── application-local.yml        # 로컬 환경 설정
├── application-dev.yml          # 개발 환경 설정
├── application-stage.yml        # 스테이지 환경 설정
├── application-prod.yml         # 운영 환경 설정
├── log4j2-local.xml            # 로컬 로그 설정
├── log4j2-dev.xml              # 개발 로그 설정
├── log4j2-stage.xml            # 스테이지 로그 설정
└── log4j2-prod.xml             # 운영 로그 설정
```

5.2 환경별 설정 파일 내용

5.2.1 공통 설정 (application.yml)

```yaml
# IITP API 설정 파일
app:
  version: @version@
  build:
    date: @build.time@
  distance-calculation:
    method: POSTGIS_SPHERE  # POSTGIS_SPHERE, POSTGIS_PLANAR, POSTGRESQL_EARTH

spring:
  profiles:
    active: local
  application:
    name: iitp-api
  config:
    import: optional:file:./application-${spring.profiles.active}.yml

# API 버전 설정
api:
  version:
    current: v1
    supported:
      - v1
  rate-limit:
    enabled: true
    capacity: 100
    time-window: 60
  stats_data:
    limit_count: 50000

# 서버 설정
server:
  port: 28080
  servlet:
    context-path: /
```

5.2.2 환경별 설정 예시

개발 환경 (application-dev.yml):

```
spring:
  datasource:
    driver-class-name: org.postgresql.Driver
    url: jdbc:postgresql://192.168.60.146:5432/iitp_dev
    username: iitp_dev
    password: ENC(1kZt5toeoCpGCSozUyzRHGr8uA3gyu4aaNvj/8SR+rXhrj41MocJzzyWkjwXozv2)
    hikari:
      pool-name: IITPHikariPool
      maximum-pool-size: 10
      minimum-idle: 5
      idle-timeout: 300000
      connection-timeout: 30000
      max-lifetime: 1200000

logging:
  config: classpath:log4j2-dev.xml
```

5.3 Jasypt를 이용한 암호화

5.3.1 암호화 방법

Gradle 태스크 사용:

```
# 암호화 실행
./gradlew encrypt --plain-text "내비밀번호" --password "암호화키" -Dfile.encoding=UTF-8

# 출력 예시
# ENC(1kZt5toeoCpGCSozUyzRHGr8uA3gyu4aaNvj/8SR+rXhrj41MocJzzyWkjwXozv2)
```

암호화 스크립트 사용:

```
# Linux/Mac
./encrypt.sh "내비밀번호" "암호화키"

# Windows
encrypt.bat "내비밀번호" "암호화키"
```

5.3.2 암호화된 값 사용

설정 파일에 적용:

```
spring:
  datasource:
    password: ENC(1kZt5toeoCpGCSozUyzRHGr8uA3gyu4aaNvj/8SR+rXhrj41MocJzzyWkjwXozv2)
```

5.3.3 복호화 환경변수 설정

서버 실행 시 환경변수 설정:

```
# 환경변수로 설정
export JASYPT_ENCRYPTOR_PASSWORD=암호화키

# 또는 실행 옵션으로 설정
java -jar iitp-api.jar -Djasypt.encryptor.password=암호화키
```

api_run.sh 스크립트에 설정:

```
#!/bin/bash
export SPRING_PROFILES_ACTIVE=dev
export JASYPT_ENCRYPTOR_PASSWORD=iitp  # 암호화 키 설정
```

6. 빌드

소스코드를 빌드하여 실행 가능한 JAR 파일을 생성합니다.

6.1 빌드 환경 확인

빌드 전 환경을 확인합니다.

```
# Gradle 래퍼 권한 설정
chmod +x gradlew

# Gradle 버전 확인
./gradlew --version

# 프로젝트 구조 확인
tree -L 2  # 또는 ls -la
```

6.2 빌드 과정

6.2.1 빌드 단계별 과정

빌드는 다음과 같은 단계로 진행됩니다:

1. **Clean 단계**: 이전 빌드 결과물 정리
2. **환경별 설정 처리**: 환경에 맞는 설정 파일만 포함 (나머지 제외)
3. **JAR 파일 생성**: 실행 가능한 JAR 파일 생성

참고: OpenAPI 문서 생성은 별도로 `copyDocsToResources` 태스크를 실행해야 합니다.

6.3 빌드 실행

6.3.1 빌드 스크립트 사용 (권장)

Linux/Mac:

```
# 환경별 빌드 (OpenAPI 문서 생성 포함)
./build.sh local    # 로컬 환경 (기본값)
./build.sh dev      # 개발 환경
./build.sh stage    # 스테이지 환경
./build.sh prod     # 운영 환경
```

참고: 빌드 스크립트는 `copyDocsToResources` 태스크를 포함하여 OpenAPI 문서도 함께 생성합니다.

6.3.2 Gradle 태스크 직접 사용

```
# 환경별 빌드 (JAR 파일만 생성)
./gradlew buildLocal   # 로컬 환경
./gradlew buildDev     # 개발 환경
./gradlew buildStage   # 스테이지 환경
./gradlew buildProd    # 운영 환경

# OpenAPI 문서 생성 (별도 실행 필요)
./gradlew copyDocsToResources
```

6.4 OpenAPI 문서 생성

6.4.1 자동 생성 과정

중요: OpenAPI 문서 생성은 **buildLocal**이 필요합니다.

`copyDocsToResources` 태스크 실행 시 자동으로 처리되는 과정:

1. **컴파일**: 소스코드 컴파일 (buildLocal 필요)
2. **서버 실행**: 로컬 프로필로 서버를 28081 포트에서 자동 실행
3. **문서 생성**: `/v3/api-docs` 엔드포인트에서 OpenAPI 문서를 `docs-dist` 디렉토리에 다운로드
4. **문서 복사**: `docs-dist` 에서 `src/main/resources/static/docs/` 로 복사
5. **그룹핑 추가**: Python 스크립트로 x-tagGroups 자동 추가

6.4.2 OpenAPI 문서 생성 사용법

```
# OpenAPI 문서 생성 및 복사 (권장)
./gradlew copyDocsToResources
```

중요 사항:

- 문서 생성 시 **로컬 프로필로 서버가 28081 포트에서 자동 실행**됩니다

- **application-local.yml** 설정이 필요합니다
- **데이터베이스 연결**이 필요할 수 있습니다 (서버 시작 시)

6.4.3 문서 그룹핑 자동 추가

OpenAPI 문서에 x-tagGroups가 자동으로 추가됩니다:

- `copyDocsToResources` 태스크 실행 시 자동으로 Python 스크립트가 실행됩니다
- `scripts/add-x-tagGroups.py` 가 OpenAPI 문서에 그룹핑 정보를 추가합니다

6.5 빌드 결과 확인

```
# 빌드 결과 확인
ls -la build/libs/

# 생성된 파일들
# - iitp-api-0.0.5-dev-20241219162047.jar

# OpenAPI 문서 확인
ls -la src/main/resources/static/docs/
# - latest.yaml
# - openapi-v0.0.4.yaml

# 문서 그룹핑 확인
grep -A 10 "x-tagGroups" src/main/resources/static/docs/latest.yaml
```

7. 배포

빌드된 JAR 파일을 서버에 배포합니다.

7.1 배포 디렉토리 준비

서비스 계정과 배포 디렉토리를 준비합니다.

```
# iitp-api 서비스 계정 생성
sudo useradd -r -s /bin/false iitp-api

# 배포 디렉토리 생성
sudo mkdir -p /home/iitp-api
sudo chown iitp-api:iitp-api /home/iitp-api
sudo chmod 755 /home/iitp-api

# 로그 디렉토리 생성
sudo mkdir -p /home/iitp-api/logs
sudo chown iitp-api:iitp-api /home/iitp-api/logs
sudo chmod 755 /home/iitp-api/logs
```

7.2 JAR 파일 배포

7.2.1 파일

```
# 빌드된 JAR 파일을 배포
 /home/iitp-api/로 배포 (iitp-api.jar)

# 파일 권한 설정
sudo chown iitp-api:iitp-api /home/iitp-api/*.jar
sudo chmod 644 /home/iitp-api/*.jar
```

확인 방법:

```
# 파일 확인
ls -la /home/iitp-api/iitp-api.jar

# 파일 내용 확인 (압축 해제)
jar -tf /home/iitp-api/iitp-api.jar | head -20
```

7.2 실행 스크립트 배포

```
# 실행 스크립트
위치 :  /home/iitp-api/
sudo chmod +x /home/iitp-api/api_run.sh
sudo chown iitp-api:iitp-api /home/iitp-api/api_run.sh
```

7.4 배포 디렉토리 구조 확인

```
# 배포 디렉토리 구조 확인
ls -la /home/iitp-api/

# 예상 결과:
# drwxr-xr-x 2 iitp-api iitp-api 4096 Oct 24 10:00 .
# drwxr-xr-x 3 root     root     4096 Oct 24 10:00 ..
# -rwxr-xr-x 1 iitp-api iitp-api 1234 Oct 24 10:00 api_run.sh
# -rw-r--r-- 1 iitp-api iitp-api 123M Oct 24 10:00 iitp-api.jar
# drwxr-xr-x 2 iitp-api iitp-api 4096 Oct 24 10:00 logs
```

8. 서버 실행 및 자동 시작 설정

8.1 수동 실행

8.1.1 실행 스크립트 확인

실행 스크립트의 내용을 확인합니다.

```
# api_run.sh 파일 내용 확인
cat /home/iitp-api/api_run.sh

# Default :
# export SPRING_PROFILES_ACTIVE=dev
# export JASYPT_ENCRYPTOR_PASSWORD=iitp
```

참고: Default 설정을 project 상황에 맞게 변경할 수 있음.
단 ***5.3 Jasypt를 이용한 암호화*** 절차에서 사용한 암호화키와 일치해야함.

8.1.2 수동 실행

```
# iitp-api 계정으로 전환
sudo su - iitp-api

# 서버 실행
cd /home/iitp-api
./api_run.sh

# 백그라운드 실행 (선택사항)
nohup ./api_run.sh > logs/iitp-api.log 2>&1 &
```

8.2 systemd 서비스 설정

8.2.1 서비스 파일 생성

```
# systemd 서비스 파일 생성
sudo cp iitp-api.service /etc/systemd/system/

# 서비스 파일 권한 설정
sudo chmod 644 /etc/systemd/system/iitp-api.service
```

8.2.2 서비스 파일 내용 확인

iitp-api.service 파일 내용:

```
[Unit]
Description=IITP API Service
After=network.target

[Service]
Type=simple
User=iitp-api
Group=iitp-api
WorkingDirectory=/home/iitp-api
Environment=INVOKED_BY_SYSTEMD=1
ExecStart=/home/iitp-api/api_run.sh
Restart=always
RestartSec=10

[Install]
WantedBy=multi-user.target
```

8.2.3 서비스 활성화 및 시작

```
# systemd 데몬 리로드
sudo systemctl daemon-reload

# 서비스 활성화 (부팅 시 자동 시작)
sudo systemctl enable iitp-api

# 서비스 시작
sudo systemctl start iitp-api

# 서비스 상태 확인
sudo systemctl status iitp-api
```

8.3 서비스 관리 명령어

```
# 서비스 시작
sudo systemctl start iitp-api

# 서비스 중지
sudo systemctl stop iitp-api

# 서비스 재시작
sudo systemctl restart iitp-api

# 서비스 상태 확인
sudo systemctl status iitp-api

# 서비스 로그 확인
sudo journalctl -u iitp-api -f

# 서비스 비활성화 (부팅 시 자동 시작 해제)
sudo systemctl disable iitp-api
```

9. 배포 후 확인 및 테스트

배포된 서버가 정상적으로 동작하는지 확인하고 테스트합니다.

9.1 서버 상태 확인

서버의 기본 상태를 확인합니다.

```
# 서비스 상태 확인
sudo systemctl status iitp-api

# 프로세스 확인
ps aux | grep iitp-api

# 포트 확인
netstat -tlnp | grep 28080
```

9.2 API 엔드포인트 테스트

9.2.1 기본 상태 확인

```
# 서버 상태 확인
curl -X GET "http://localhost:28080/api/v1/comm/health"

# 서버 버전 확인
curl -X GET "http://localhost:28080/api/v1/comm/version"
```

9.2.2 OpenAPI 문서 확인

```
# OpenAPI 문서 확인
curl -X GET "http://localhost:28080/v3/api-docs"

# Swagger UI 확인 (브라우저에서)
# http://localhost:28080/swagger-ui.html
```

9.3 로그 확인

```
# 실시간 로그 확인
tail -f /home/iitp-api/logs/iitp-api.log

# 최근 로그 확인
tail -100 /home/iitp-api/logs/iitp-api.log

# 에러 로그 확인
grep -i error /home/iitp-api/logs/iitp-api.log
```

10. 로그 관리

서버 로그 파일의 위치와 관리 방법을 설명합니다.

10.1 로그 파일 위치

로그 파일의 위치와 권한을 확인합니다.

```
# 로그 파일 위치
/home/iitp-api/logs/iitp-api.log

# 로그 파일 권한 확인
ls -la /home/iitp-api/logs/
```

10.2 로그 로테이션 설정

```
# logrotate 설정 파일 생성
sudo cp iitp-api-console.log.logrotate /etc/logrotate.d/iitp-api

# logrotate 설정 파일 내용 확인
cat /etc/logrotate.d/iitp-api
```

logrotate 설정 파일 내용:

```
/home/iitp-api/logs/iitp-api.log {
    daily
    rotate 30
    compress
    delaycompress
    missingok
    notifempty
    create 644 iitp-api iitp-api
    postrotate
        /bin/kill -USR1 `cat /var/run/iitp-api.pid 2> /dev/null` 2> /dev/null || true
    endscript
}
```

10.3 로그 분석

```
# 최근 에러 로그 확인
grep -i error /home/iitp-api/logs/iitp-api.log | tail -20

# 특정 시간대 로그 확인
grep "2024-10-24 10:" /home/iitp-api/logs/iitp-api.log

# API 호출 로그 확인
grep "GET\|POST\|PUT\|DELETE" /home/iitp-api/logs/iitp-api.log | tail -20
```

11. 문제 해결

서버 운영 중 발생할 수 있는 문제들과 해결 방법을 설명합니다.

11.1 서비스 시작 실패

11.1.1 서비스 상태 확인

서비스가 시작되지 않는 경우 상태를 확인합니다.

```
# 서비스 상태 확인
sudo systemctl status iitp-api

# 서비스 로그 확인
sudo journalctl -u iitp-api -n 50
```

11.1.2 일반적인 해결 방법

```
# 서비스 재시작
sudo systemctl restart iitp-api

# systemd 데몬 리로드
sudo systemctl daemon-reload

# 서비스 파일 권한 확인
ls -la /etc/systemd/system/iitp-api.service
```

11.2 데이터베이스 연결 문제

11.2.1 연결 테스트

```
# 데이터베이스 연결 테스트
psql -h [호스트주소] -U [사용자명] -d [데이터베이스명]

# 네트워크 연결 테스트
telnet [호스트주소] 5432
```

11.2.2 설정 확인

```
# JAR 파일 내 설정 확인
jar -tf /home/iitp-api/iitp-api.jar | grep application

# 환경변수 확인
echo $SPRING_PROFILES_ACTIVE
echo $JASYPT_ENCRYPTOR_PASSWORD
```

11.3 포트 충돌 문제

```
# 포트 사용 확인
netstat -tlnp | grep 28080

# 프로세스 확인
ps aux | grep java

# 포트 변경 (필요시)
# application.yml에서 server.port 수정
```

11.4 메모리 부족 문제

```
# 메모리 사용량 확인
free -h

# Java 힙 메모리 설정 (필요시)
# JVM 옵션 추가: -Xms512m -Xmx1024m
```

---

12. 참고 사항

12.1 주요 파일 위치

파일/디렉토리	위치	설명
JAR 파일	`/home/iitp-api/iitp-api.jar`	실행 파일
실행 스크립트	`/home/iitp-api/api_run.sh`	서버 실행 스크립트
서비스 파일	`/etc/systemd/system/iitp-api.service`	systemd 서비스 설정
로그 파일	`/home/iitp-api/logs/iitp-api.log`	애플리케이션 로그
설정 파일	JAR 파일 내부	환경별 설정 파일

12.2 주요 명령어

```
# 서비스 관리
sudo systemctl start|stop|restart|status iitp-api

# 로그 확인
tail -f /home/iitp-api/logs/iitp-api.log

# API 테스트
curl -X GET "http://localhost:28080/api/v1/comm/health"
```

12.3 문제 해결 체크리스트

- ☐ 서비스 상태 확인 (`systemctl status iitp-api`)
- ☐ 로그 파일 확인 (`tail -f /home/iitp-api/logs/iitp-api.log`)
- ☐ 포트 확인 (`netstat -tlnp | grep 28080`)
- ☐ 데이터베이스 연결 확인
- ☐ 환경변수 확인
- ☐ 파일 권한 확인